U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEIC 2º Ano

Novembro 2018

ALGORITMOS E ESTRUTURAS DE DADOS

Trabalho Prático – Parte 1

Tema 1 - Transporte Escolar

Turma 5 – Grupo 7:

João Praça - up201704748@fe.up.pt

Leonor Sousa - up201705377@fe.up.pt

Sílvia Rocha - up201704684@fe.up.pt

Prof.ª Ana Paula Cunha da Rocha

Prof. Luís Paulo Gonçalves dos Reis

Índice

Descrição do Tema

O tema abordado neste projeto tem por objetivo implementar um sistema de gestão adequado e eficiente para empresas de transporte escolar.

O conceito de transporte escolar surgiu com a pretensão de simplificar o quotidiano das famílias que a ele aderem dado que promovem o transporte coletivo de alunos e funcionários desde as suas casas até às respetivas escolas e vice-versa. Este sistema apresenta como principal vantagem a melhoria na mobilidade dos seus utentes, dado que as pessoas passam a ter um transporte que as leva até ao seu destino sem necessitarem de efetuar qualquer tipo de transbordo.

Para este tipo de transporte, é utilizado um método muito semelhante ao sistema ANDANTE. A região onde a empresa opera é dividida em várias zonas, sendo que cada utente e cada escola terão uma zona associada à sua localização. Cada utente tem uma mensalidade definida, que se baseia numa tabela de preços por zonas percorridas.

Estas empresas fornecem ainda uma vertente de serviço ocasional que visa, essencialmente, a realização de serviços pontuais ao longo do tempo. Estes serviços são muitas vezes contratados, por exemplo, para a deslocação de alunos e funcionários aquando da realização de visitas de estudo.

Tendo em conta que cada empresa pode surgir associada a várias escolas é fundamental efetuar uma gestão inteligente e cautelosa dos recursos limitados que estão à disposição da empresa de forma a maximizar o número de utentes que consegue aceitar e, consequentemente, os lucros mensais da empresa. É, portanto, nesse sistema otimizado de gestão que a nossa versão de implementação pretende incidir.

Descrição da Solução Implementada

Dado o problema acima descrito implementamos, de forma modular, a solução que, do nosso ponto de vista, dá uma melhor resposta tendo em conta a complexidade de dados com que é necessário lidar. Deste modo, implementamos as seguintes classes:

- Empresa
- Escola
- Utente
  - ✓ Crianca
  - ✓ Funcionario
- Veículo
  - ✓ ServicoRegular
  - ✓ ServicoEspecial
- Date

Em cada função desenvolvida tentámos ainda obter a maior redundância possível fazendo o tratamento de todas as exceções que, na nossa perspetiva, poderiam surgir ao longo da execução. Para isto tivemos em conta o input e dados que seriam alterados em cada função. Deste modo, criámos as seguintes classes para tratamento de exceções(todas elas derivadas da classe exception da STL):

- ZonaInexistente;
- UtenteInexistente;
- UtenteRepetido;
- VeiculoInexistente;
- VeiculoIndisponivel;
- VeiculoRepetido;
- DataInvalida;
- RemocaoVeiculoImpossivel;

Do ponto de vista do utilizador, existe abstração total da forma como a estrutura foi programada dado que este apenas interage diretamente com a interface criada para o efeito. Esta interface é, portanto, a ponte de ligação entre o utilizador e todas as funcionalidades implementadas.

Na criação de uma nova empresa de transporte escolar existem, desde logo, alguns fatores importantes que é necessário definir, entre eles como e quanto cobrar aos utentes que subscrevem cada um dos serviços que a empresa disponibiliza.

Neste sentido, a empresa possui 2 estruturas de dados fundamentais à gestão dos serviços regulares: um vetor de valores do tipo double e um map cujo primeiro elemento é um par de 2 zonas e o segundo elemento é o número de zonas envolvidas no percurso entre as zonas do primeiro elemento. Será a estas estruturas que o programa vai recorrer sempre que for necessário calcular o valor que um determinado utente tem a pagar. Primeiramente, tendo em conta a zona de residência do utente e a zona da sua escola, irá percorrer o map para obter o número de zonas a cobrar. De seguida, irá aceder ao elemento do vetor cujo índice é o número

de zonas obtido obtendo o valor total a pagar pelo utente (este valor não considera ainda o sistema de descontos abordado na lista de casos de uso mais à frente neste documento, esse cálculo é feito posteriormente).

Para além destas estruturas, para as situações de serviço ocasional, é ainda definido o valor a cobrar por lugar. Este valor multiplicado pela capacidade do veículo define o valor a cobrar por cada quilómetro efetuado. Deste modo, para obter o valor total a cobrar por um determinado serviço teremos ainda de multiplicar esse valor pelo número de quilómetros percorridos.

Sempre que um novo utente quer subscrever os serviços desta empresa é alocado a um dos veículos do tipo ServicoRegular tendo em conta, como critério de alocação, a escola a que esse utente pertence. Assim, se já existir um veículo alocado à escola do utente a alocar e a sua capacidade ainda não tiver sido atingida o utente será alocado a este veículo, caso contrário, caso existam veículos de serviço regular disponíveis, um novo veículo será alocado à escola do utente e, por sua vez, o utente será alocado a esse veículo. Na função alocaUtente() optámos por efetuar overload de forma a mais facilmente distinguir se o utente a alocar é uma criança ou um funcionário.

No caso dos serviços ocasionais é utilizada a função alocaServico(). Esta função começa por ordenar o vetor de veículos (como descrito no ponto 5 da listagem de casos de uso mais à frente neste documento). Tendo em conta essa ordenação, o veículo que será alocado a um determinado serviço ocasional será o primeiro veículo do tipo ServicoEspecial que esteja disponível na data do serviço e que possua uma capacidade igual ou superior ao número de pessoas que irão usufruir do serviço (como o vetor está ordenado por ordem de capacidades o veículo atribuído terá sempre a capacidade mais próxima do número de pessoas possível).

Diagrama UML

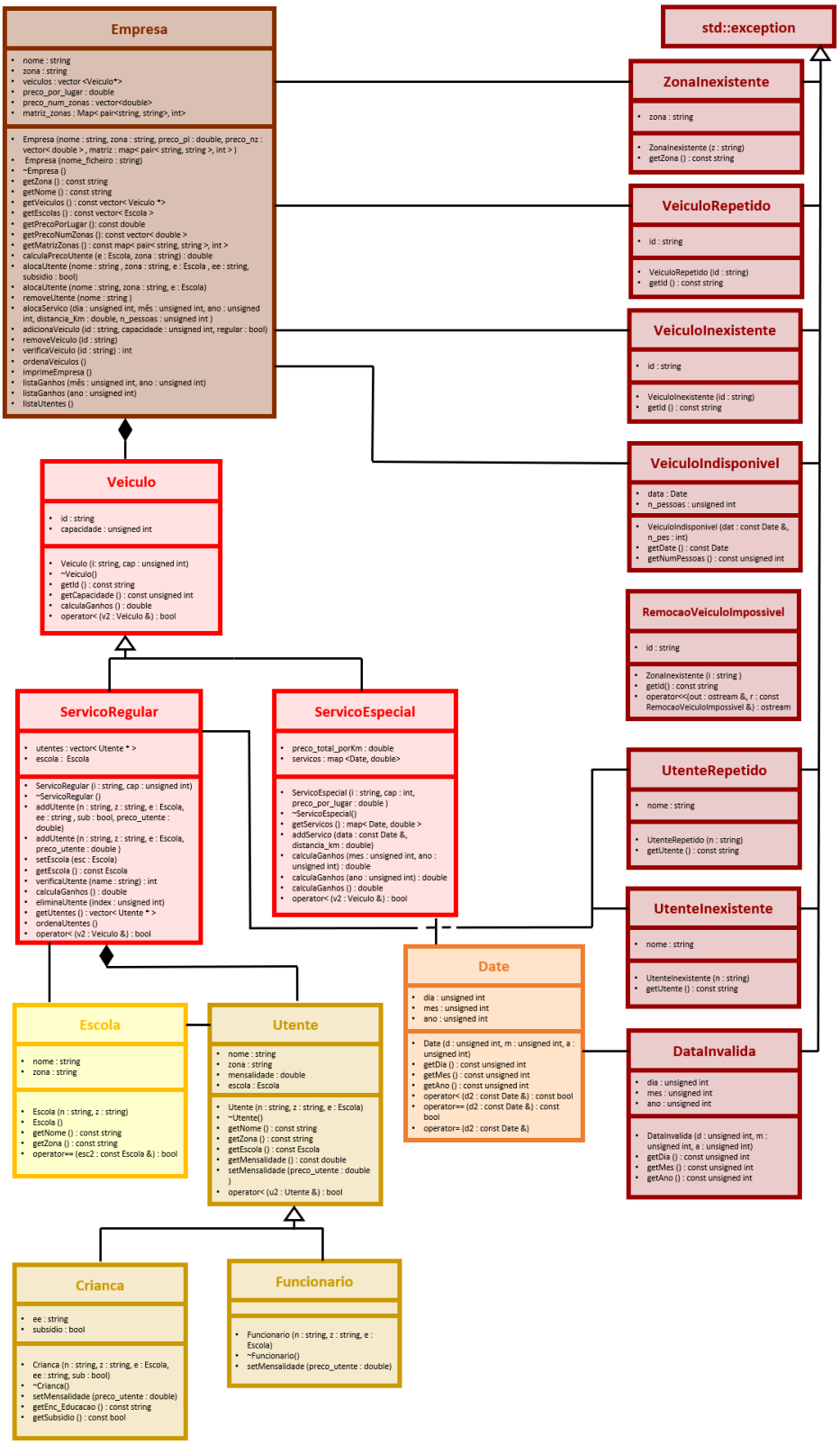

Figura 1 - Diagrama UML de Classes

Nota: Será enviado, junto com o relatório, o diagrama para melhor visualização do mesmo.

Lista de Casos de Utilização

1. A possibilidade de guardar e "carregar" uma Empresa

Uma empresa pode ser criada a partir da leitura de um ficheiro previamente criado, no formato adequado (ver figura). Para permitir a execução desta funcionalidade foram definidos o operador<< para a classe Empresa e a função imprimeEmpresa() que coloca toda a informação relevante da empresa num documento. Isto permite que toda essa informação possa ser recuperada em utilizações futuras do programa através do construtor que realiza o processo inverso, isto é, que lê o ficheiro linha a linha de modo a construir uma empresa com os dados extraídos.

```
(início de ficheiro)
<Nome da Empresa>
<Zona da Empresa>
<Preco por Lugar>

<Preco_num_zonas[0]>
<Preco_num_zonas[1]>
<...>

<zona 1> <zona2> <num_zonas_intermedias>
<...>

<ID do VeiculoReg 1>
<Capacidade do VeiculoReg 1>
<Nome da escola do VeiculoReg 1> (se não tiver escola atribuída, esta linha está vazia)
<Zona da escola do VeiculoReg 1> (se não tiver escola atribuída, esta linha está vazia)

<Nome da Criança1 do VeiculoReg1>
<Zona da Criança1 do VeiculoReg1>
<Mensalidade da Criança1 do VeiculoReg1>
<EE da Criança1 do VeiculoReg1>
<Subsidio da Criança1 do VeiculoReg1(S->true, N->false)>

<Nome do Funcionario1 do VeiculoReg1>
<Zona do Funcionario do VeiculoReg1>
<Mensalidade do Funcionario do VeiculoReg1>

<...outros utentes...>


<...outros veiculosRegulares...>


.
<ID do VeiculoEsp 1>
<Capacidade do VeiculoEsp 1>
<dia>.<mes>.<ano> <ganho>
<...>

<...outros veiculosEspeciais...>


(fim de ficheiro)
```

Figura 2 - Formato de Ficheiro utilizado para "guardar" a empresa.

2. Gestão da frota de autocarros da Empresa

Através das funções adicionaVeiculo() e removeVeiculo(), é possível, respetivamente, adicionar e remover um veículo. A função adicionaVeiculo() recebe, como um dos seus argumentos, um booleano que indica se o veículo que está a ser adicionado terá como função serviços regulares ou ocasionais. Já o removeVeiculo() apenas precisa de receber como argumento o ID representativo do veículo a remover. Para além destas funcionalidades temos ainda as funções alocaServico() e alocaUtente(). A função alocaServico() aloca, a um veículo do tipo ServicoEspecial, uma determinada deslocação numa determinada data tornando-o indisponível para outros serviços nessa mesma data. A função alocaUtente() tem por objetivo alocar cada criança/funcionário a um determinado veículo, do tipo ServicoRegular, tendo em conta a escola que frequentam/ em que trabalham.

3. Gestão dos utentes da Empresa

Através das funções alocaUtente() e removeUtente é permitido ao utilizador alterar a sua carteira de clientes sempre que um cliente novo surja ou algum dos atuais tenha de ser removido. Na ajuda a esta gestão o utilizador pode ainda visualizar todos os clientes da empresa através de uma das opções de listagem que será abordada num ponto mais à frente neste documento.

À semelhança do que acontece na realidade dos transportes que utilizamos no nosso quotidiano, foi implementado um sistema de descontos para os utentes da empresa tendo em conta a sua situação socioeconómica. Deste modo, um Utente do tipo Crianca obtém automaticamente 25% de desconto sobre o valor total da mensalidade. Caso o booleano associado a esse mesmo utente indique que este aluno tem direito a apoios sociais(subsídio) o desconto aumenta até aos 60%. No caso do Utente ser do tipo Funcionário, paga a totalidade da mensalidade.

4. Listagens de informações possivelmente relevantes da Empresa

Foram implementados três tipos diferentes de listagens:uma listagem geral e duas mais específicas. A primeira já foi abordada no ponto 1 desta secção do relatório consistindo na apresentação de toda a informação relevante da empresa, isto é, os dados da empresa, os seus veículos e utentes. Quanto às listagens mais específicas, cada uma delas associa-se a cada uma das funções: listaGanhos() e listaUtentes(). A primeira destas apresenta, no ecrã, os ganhos, relativos a um determinado período de tempo, de cada veículo da frota da empresa seguidos do valor total nesse mesmo período. Nesta função decidimos utilizar novamente overloading de modo a permitir ver os ganhos de um determinado mês ou de um determinado ano consoante as necessidades do utilizador. A função listaUtentes() apresenta no ecrã uma lista de todos os utentes associados à empresa.

5. Ordenação dos veículos

É possível, através da função ordenaVeiculos() ordenar os veículos pertencentes à frota da empresa. Nesta ordenação o primeiro critério utilizado é qual o tipo de veículo. Deste modo, em primeiro lugar são colocados todos os veículos do tipo ServicoRegular e só de seguida são colocados os veículos do tipo ServicoEspecial. Os veículos do tipo ServicoRegular são ordenados

por ordem crescente de ganhos e, no caso de dois veículos possuírem os mesmos ganhos mensais, por ordem alfabética de id. Os veículos do tipo ServicoEspecial são ordenados por ordem decrescente de capacidade. No caso de dois veículos regulares possuírem a mesma capacidade é aplicado o mesmo critério utilizado nos veículos regulares.

Aspetos sobre a Realização do Trabalho

Numa reunião inicial, o trabalho foi esboçado e foram definidas as principais classes e subclasses, assim como os atributos e métodos respetivos. Foram ainda definidos os tipos de dados usados para armazenamento de informação e também a divisão de trabalho entre os três membros.

Numa segunda reunião, o código dos três elementos foi agregado, foram corrigidas algumas incoerências e acrescentados alguns pormenores que nos pareceram fulcrais.

Para facilitar a organização de ideias que fossem surgindo e problemas que fossem necessários corrigir, foi criado um ficheiro partilhado que continha três listas:"To Vote", "To Do" e "Já Implementado", sendo que na primeira eram registadas ideias que poderiam contribuir para o projeto, e que eram votadas pelos outros membros consoante a sua pertinência; caso a ideia fosse aprovada, passaria para a lista "To Do", que continha problemas que precisavam de ser resolvidos/ sugestões que ainda deveriam ser implementadas; após a ideia ser implementada ou o problema resolvido, o tópico era movido para a lista "Já Implementado", apenas para efeitos de registo.

No decorrer do trabalho, foram realizadas ainda mais algumas reuniões, sempre que considerámos adequado.

Ao longo da realização do trabalho, foram surgindo vários problemas que tentámos sempre resolver da forma mais correta e eficaz possível. As principais dificuldades com que nos deparámos consistiram na implementação de alguns destrutores e/ou funções virtuais e no tratamento de informação dos veículos dos quais não conhecíamos o tipo. A solução encontrada para este problema passou pelo uso de dynamic_cast. No entanto, em retrospetiva, concluímos que o seu uso acabou por se alastrar mais do que o desejável e aconselhável.